ITC 6000
Database Management
SQL Murder Mystery Lab

Professor Neil Clauson

n.clauson@northeastern.edu

Overview

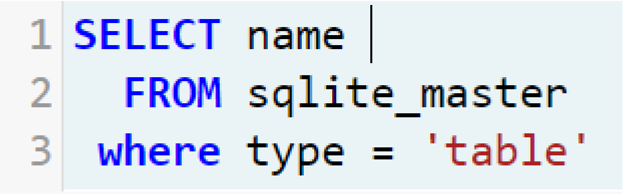
- Practice SQL commands to
  - Retrieve data
  - Filter and Sort
  - Query multiple tables through JOINs
  - Investigate and analyze data
- Perform labs using your lab environment.

SQL Murder Mystery

- A crime has taken place and the detective needs your help.
- The detective gave you the crime scene report, but you somehow lost it.
- You vaguely remember that the crime was a **murder** that occurred sometime on **Jan.15, 2018** and that it took place in **SQL City**.
- Start by retrieving the corresponding crime scene report from the police department’s database.

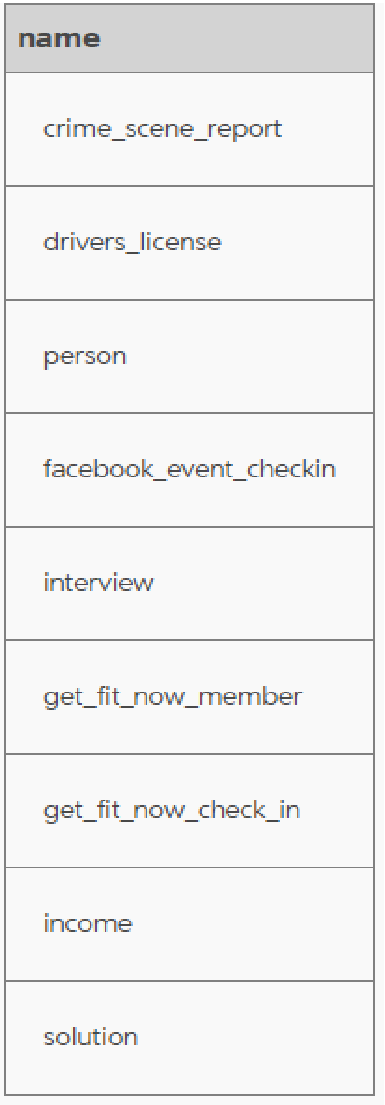
Step 1 – Get familiar with the data (1)

- Use the master schema to understand all of the tables within the database



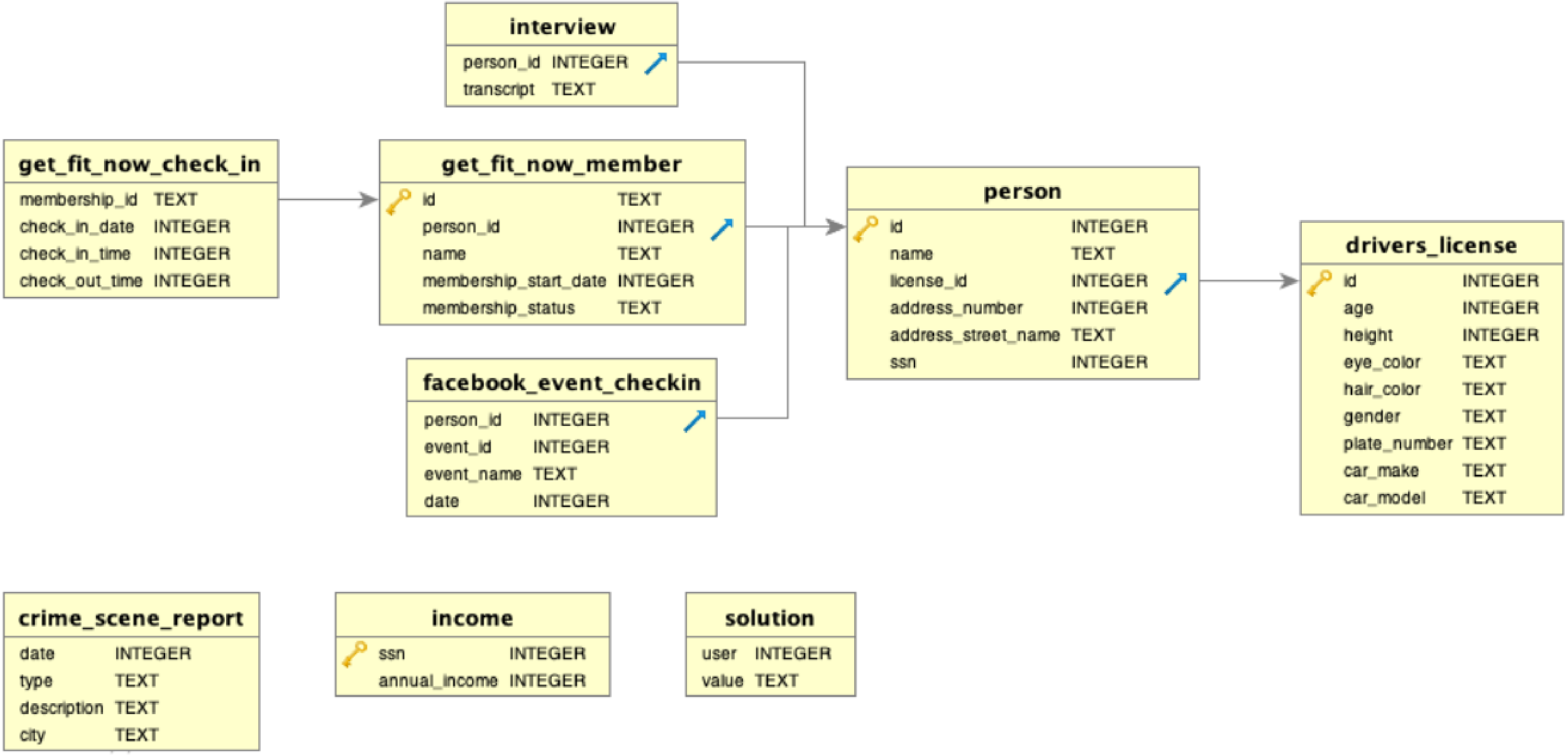
```sql
SELECT name
  FROM sqlite_master
 where type = 'table'
```

- Exact syntax depends on the type of database server



name
crime_scene_report
drivers_license
person
facebook_event_checkin
interview
get_fit_now_member
get_fit_now_check_in
income
solution

ERD Diagram



interview

person_id	INTEGER	↗
transcript	TEXT	

get_fit_now_check_in

membership_id	TEXT
check_in_date	INTEGER
check_in_time	INTEGER
check_out_time	INTEGER

get_fit_now_member

🔑	id	TEXT	
	person_id	INTEGER	↗
	name	TEXT	
	membership_start_date	INTEGER	
	membership_status	TEXT	

person

🔑	id	INTEGER	
	name	TEXT	
	license_id	INTEGER	↗
	address_number	INTEGER	
	address_street_name	TEXT	
	ssn	INTEGER	

drivers_license

🔑	id	INTEGER
	age	INTEGER
	height	INTEGER
	eye_color	TEXT
	hair_color	TEXT
	gender	TEXT
	plate_number	TEXT
	car_make	TEXT
	car_model	TEXT

facebook_event_checkin

person_id	INTEGER	↗
event_id	INTEGER	
event_name	TEXT	
date	INTEGER	

crime_scene_report

date	INTEGER
type	TEXT
description	TEXT
city	TEXT

income

🔑	ssn	INTEGER
	annual_income	INTEGER

solution

user	INTEGER
value	TEXT

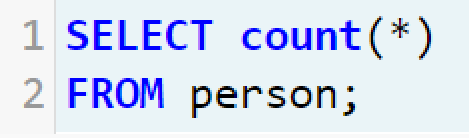
Notes: This database is design for learning purposes, and is not fully optimized (1NF, 2NF, etc)

Step 1 – Get familiar with the data (2)

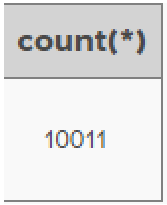
- Use table definitions to understand data types and relationships

Step 1 – Get familiar with the data (3)

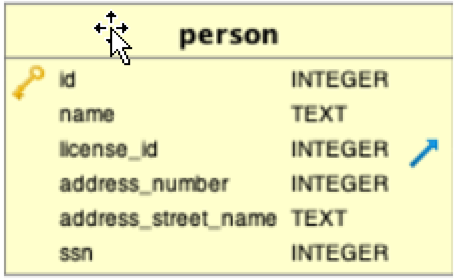
- Understand the quantity of data
- Answers “how many rows per table”



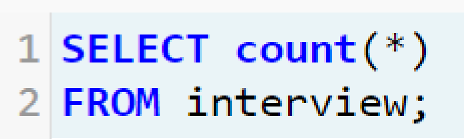
```sql
SELECT count(*)
FROM person;
```



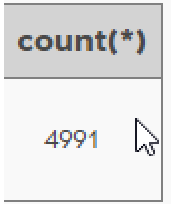
count(*)
10011



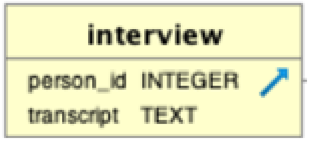
person	
id	INTEGER
name	TEXT
license_id	INTEGER
address_number	INTEGER
address_street_name	TEXT
ssn	INTEGER



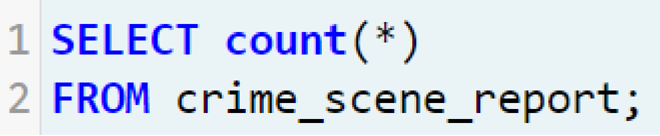
```sql
SELECT count(*)
FROM interview;
```



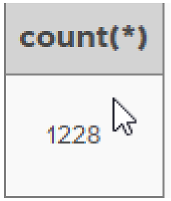
count(*)
4991



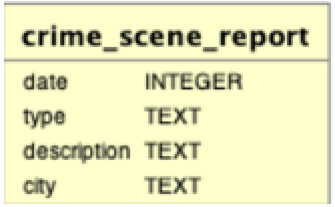
interview	
person_id	INTEGER
transcript	TEXT



```sql
SELECT count(*)
FROM crime_scene_report;
```

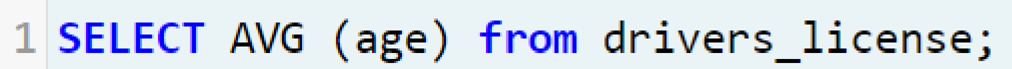


count(*)
1228

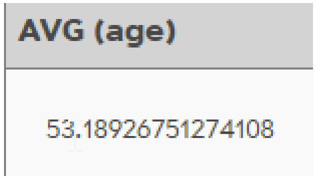


crime_scene_report	
date	INTEGER
type	TEXT
description	TEXT
city	TEXT

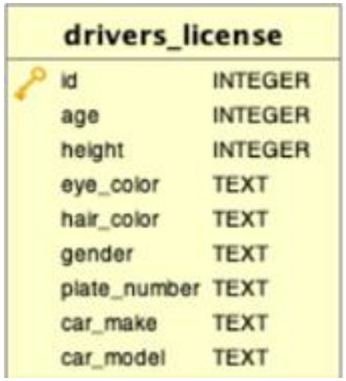
Aggregation



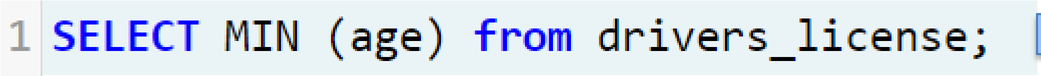
```sql
SELECT AVG (age) from drivers_license;
```



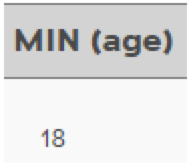
AVG (age)
53.18926751274108



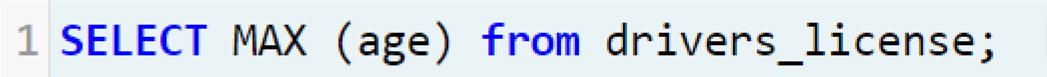
drivers_license	
id	INTEGER
age	INTEGER
height	INTEGER
eye_color	TEXT
hair_color	TEXT
gender	TEXT
plate_number	TEXT
car_make	TEXT
car_model	TEXT



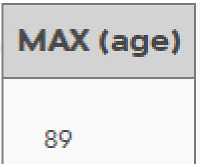
```sql
SELECT MIN (age) from drivers_license;
```



MIN (age)
18



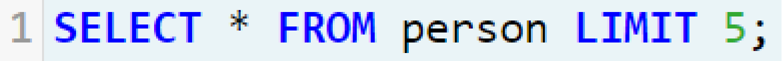
```sql
SELECT MAX (age) from drivers_license;
```



MAX (age)
89

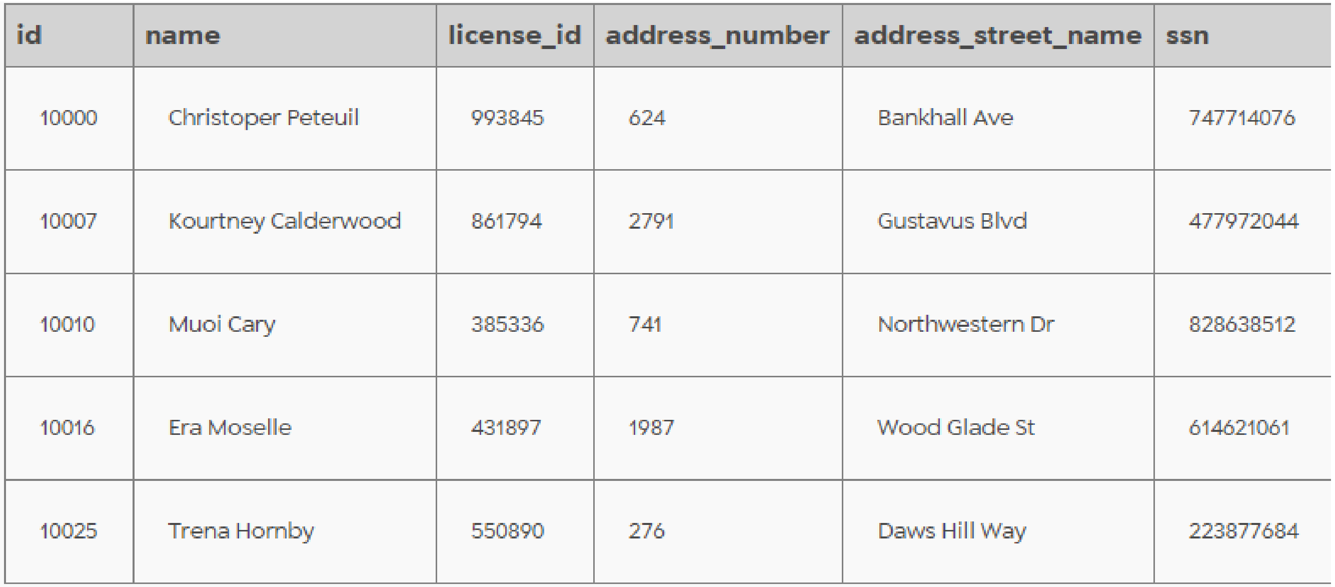
Step 1 – Get familiar with the data (3)

- Explore samples of the data:



```sql
SELECT * FROM person LIMIT 5;
```

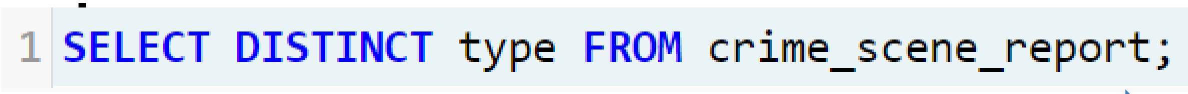
- * = wildcard. Selects all fields
- LIMIT = total number of rows returned



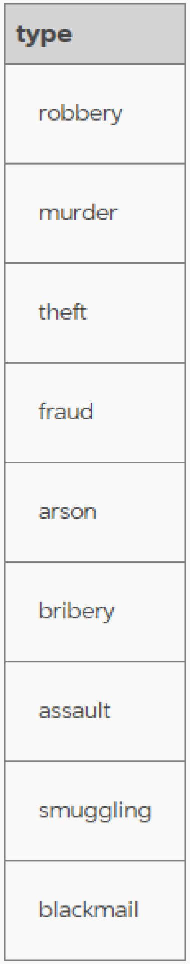
id	name	license_id	address_number	address_street_name	ssn
10000	Christoper Peteuil	993845	624	Bankhall Ave	747714076
10007	Kourtney Calderwood	861794	2791	Gustavus Blvd	477972044
10010	Muoi Cary	385336	741	Northwestern Dr	828638512
10016	Era Moselle	431897	1987	Wood Glade St	614621061
10025	Trena Hornby	550890	276	Daws Hill Way	223877684

Step 1 – Get Familiar with the Data (4)

- Understand the range of possible values within your



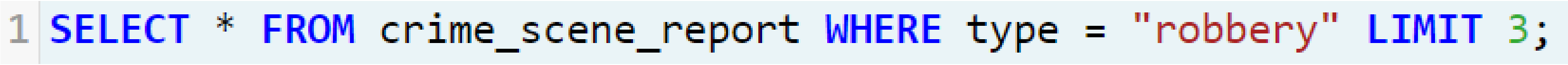
```
SELECT DISTINCT type FROM crime_scene_report;
```



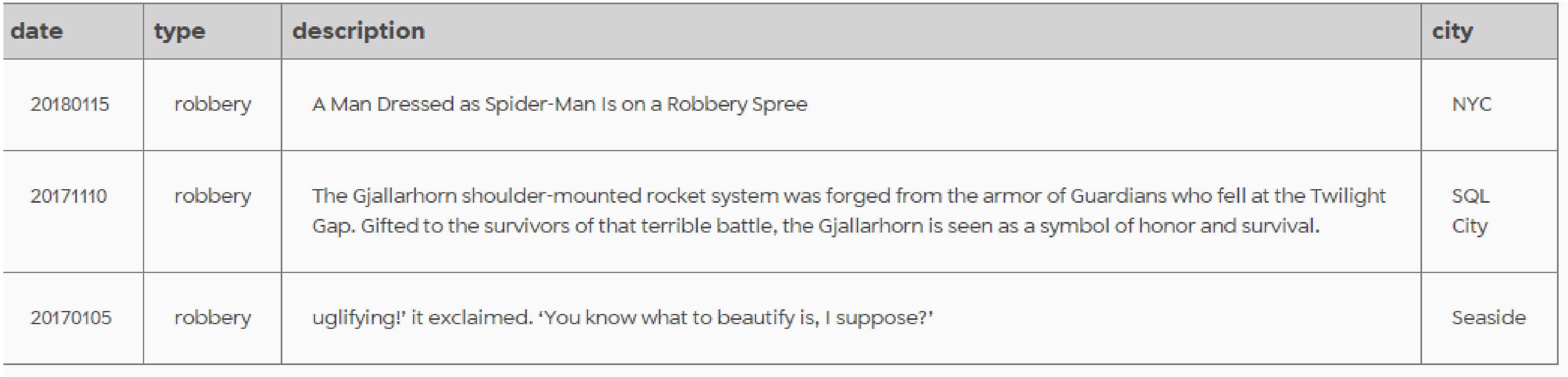
type
robbery
murder
theft
fraud
arson
bribery
assault
smuggling
blackmail

Filter Results to Specific Criteria

- SQL WHERE clause

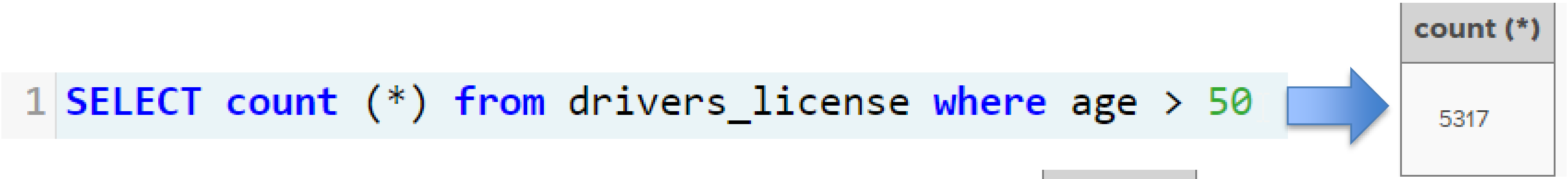


```sql
SELECT * FROM crime_scene_report WHERE type = "robbery" LIMIT 3;
```



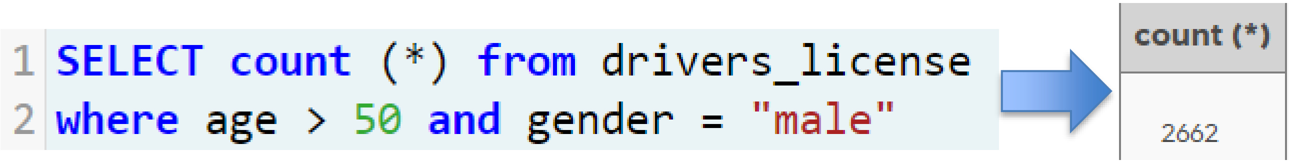
date	type	description	city
20180115	robbery	A Man Dressed as Spider-Man Is on a Robbery Spree	NYC
20171110	robbery	The Gjallarhorn shoulder-mounted rocket system was forged from the armor of Guardians who fell at the Twilight Gap. Gifted to the survivors of that terrible battle, the Gjallarhorn is seen as a symbol of honor and survival.	SQL City
20170105	robbery	uglifying!' it exclaimed. 'You know what to beautify is, I suppose?'	Seaside

Complex WHERE clauses



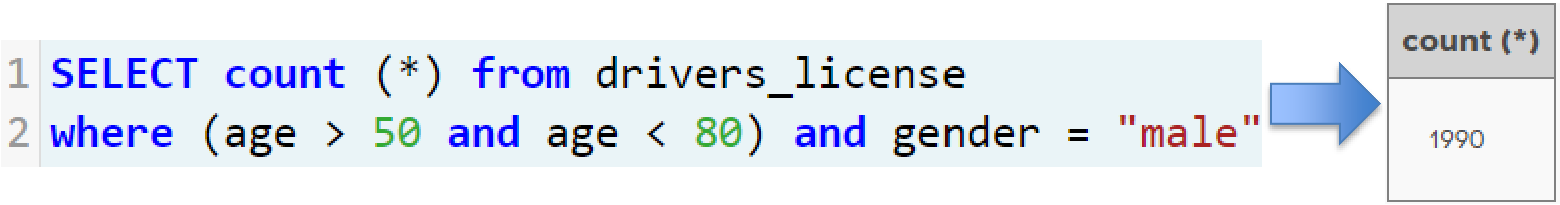
```
SELECT count (*) from drivers_license where age > 50
```

count (*)
5317



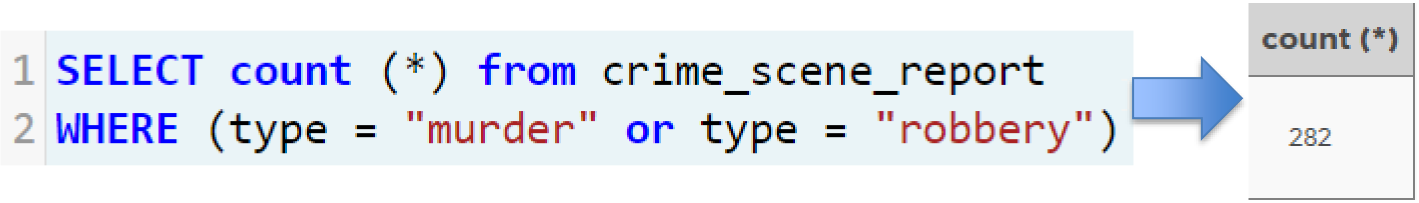
```
SELECT count (*) from drivers_license
where age > 50 and gender = "male"
```

count (*)
2662



```
SELECT count (*) from drivers_license
where (age > 50 and age < 80) and gender = "male"
```

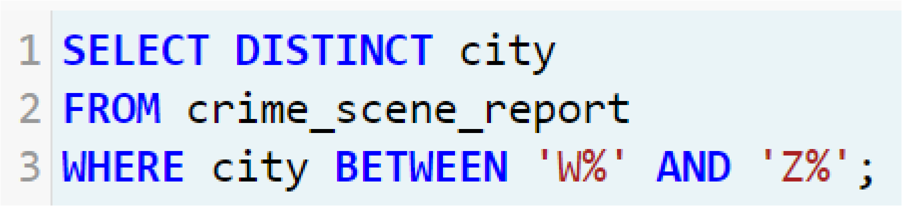
count (*)
1990



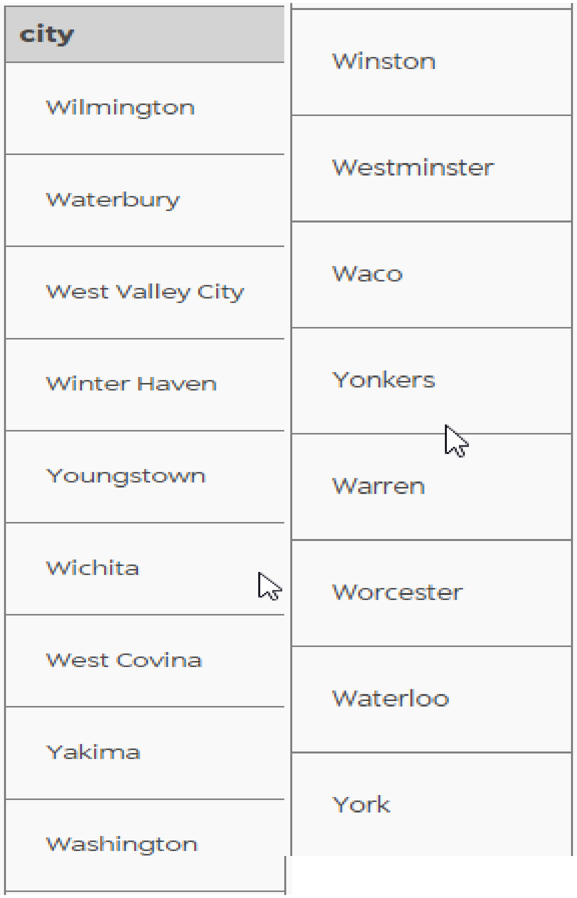
```
SELECT count (*) from crime_scene_report
WHERE (type = "murder" or type = "robbery")
```

count (*)
282

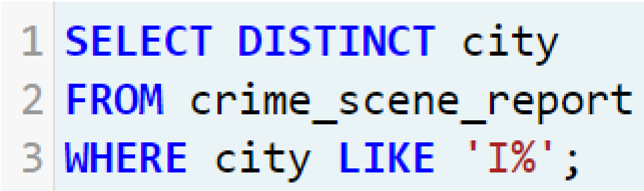
Other Comparisons



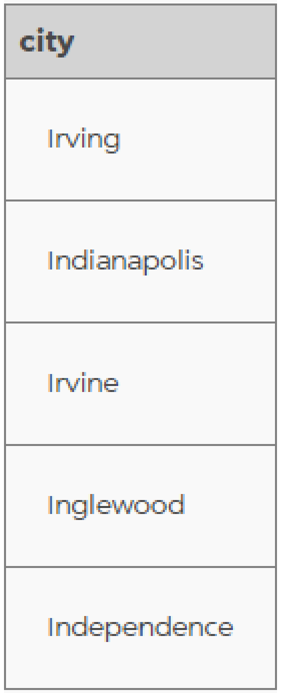
```sql
SELECT DISTINCT city
FROM crime_scene_report
WHERE city BETWEEN 'W%' AND 'Z%';
```



city
Wilmington
Waterbury
West Valley City
Winter Haven
Youngstown
Wichita
West Covina
Yakima
Washington
Winston
Westminster
Waco
Yonkers
Warren
Worcester
Waterloo
York

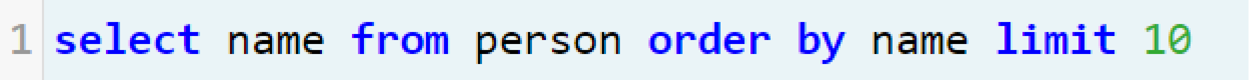


```sql
SELECT DISTINCT city
FROM crime_scene_report
WHERE city LIKE 'I%';
```

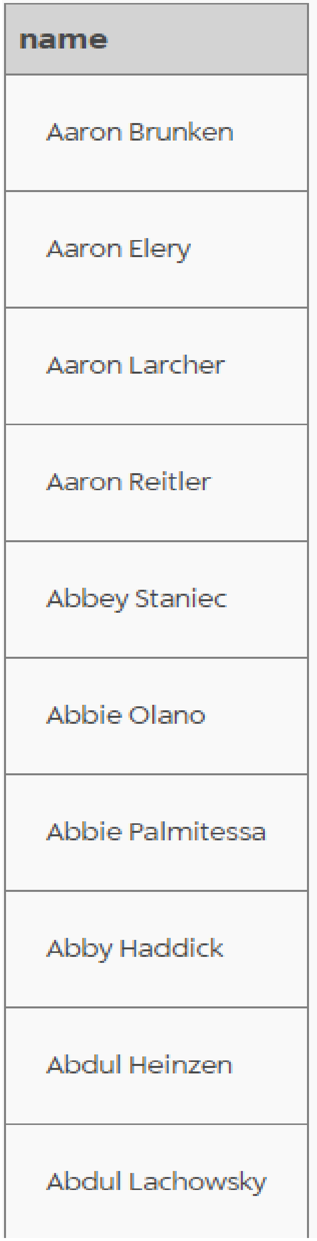


city
Irving
Indianapolis
Irvine
Inglewood
Independence

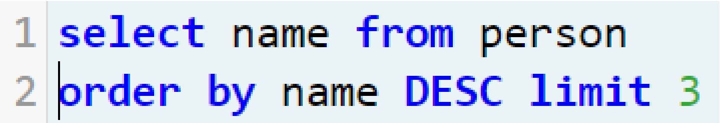
Order By



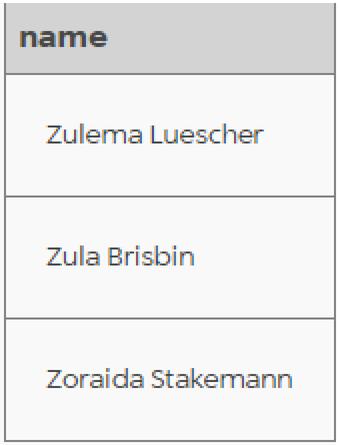
```sql
select name from person order by name limit 10
```



name
Aaron Brunken
Aaron Elery
Aaron Larcher
Aaron Reitler
Abbey Staniec
Abbie Olano
Abbie Palmitessa
Abby Haddick
Abdul Heinzen
Abdul Lachowsky

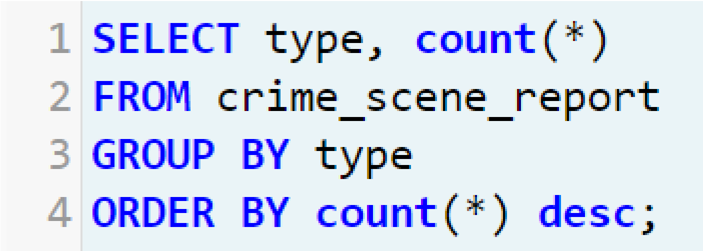


```sql
select name from person
order by name DESC limit 3
```

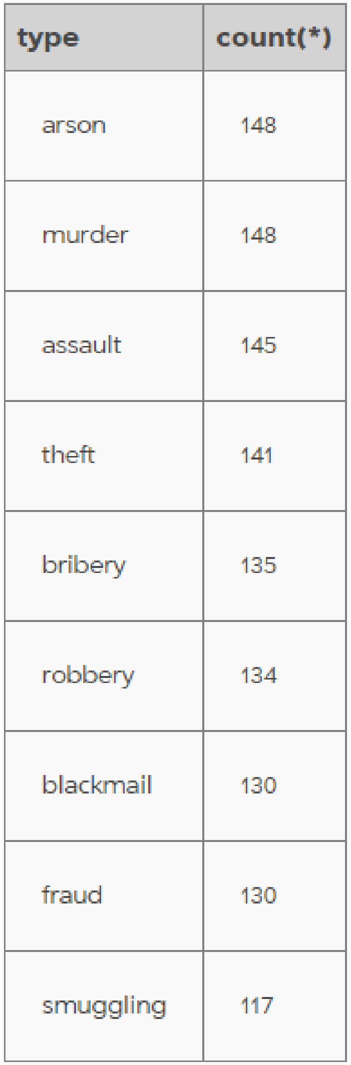


name
Zulema Luescher
Zula Brisbin
Zoraida Stakemann

Aggregating with Group By

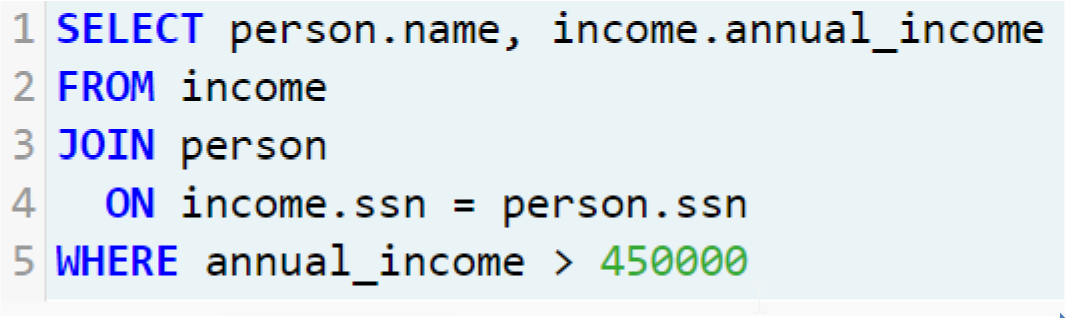


```sql
SELECT type, count(*)
FROM crime_scene_report
GROUP BY type
ORDER BY count(*) desc;
```

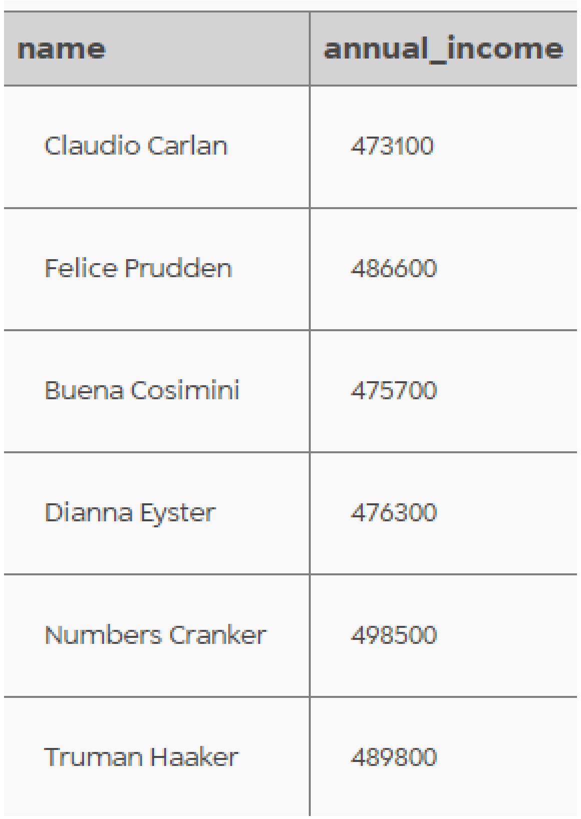


type	count(*)
arson	148
murder	148
assault	145
theft	141
bribery	135
robbery	134
blackmail	130
fraud	130
smuggling	117

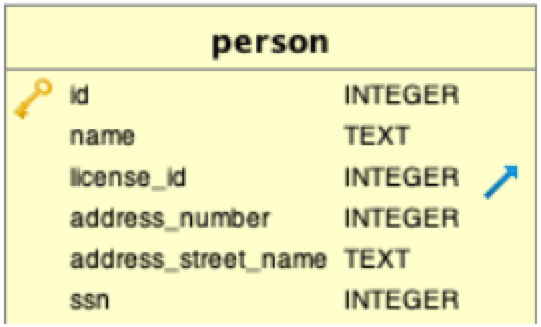
JOINs



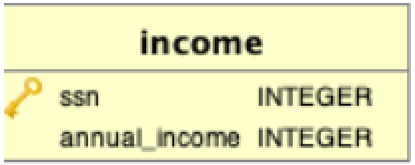
```sql
SELECT person.name, income.annual_income
FROM income
JOIN person
  ON income.ssn = person.ssn
WHERE annual_income > 450000
```



name	annual_income
Claudio Carlan	473100
Felice Prudden	486600
Buena Cosimini	475700
Dianna Eyster	476300
Numbers Cranker	498500
Truman Haaker	489800



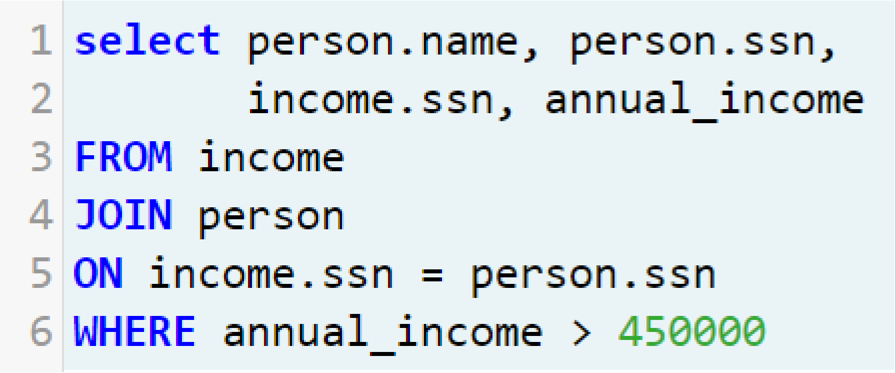
person			
🔑	id	INTEGER	
	name	TEXT	
	license_id	INTEGER	↗
	address_number	INTEGER	
	address_street_name	TEXT	
	ssn	INTEGER	



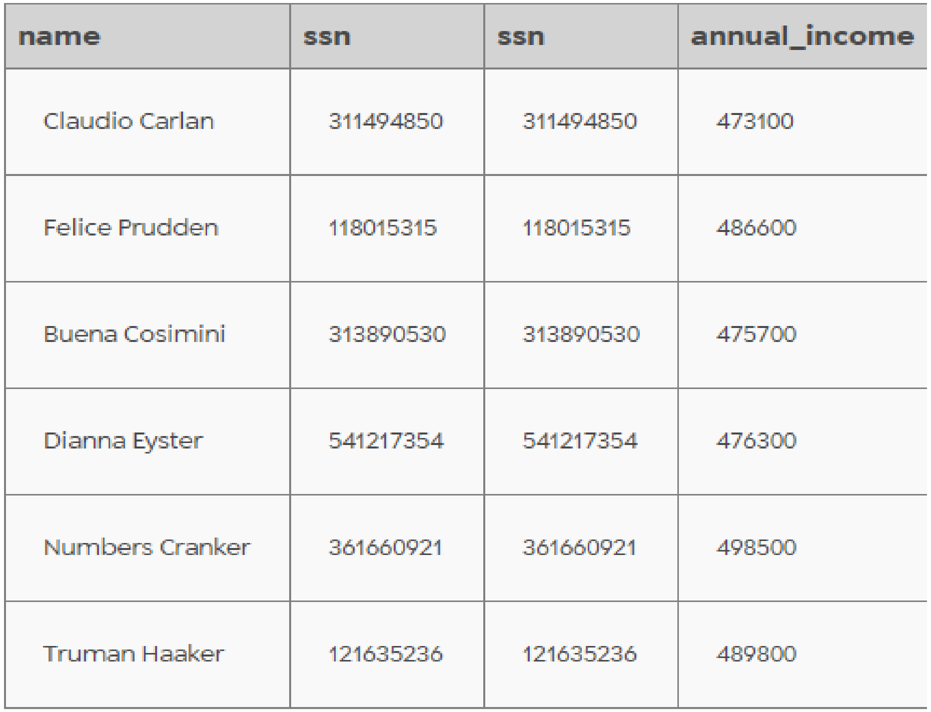
income		
🔑	ssn	INTEGER
	annual_income	INTEGER

Why does this work?

- Because the two tables share a common field (in this case, ‘ssn’)
- Shouldn’t matter which is FROM and which is JOIN
- Can be combined with all of the previous SQL syntax
- Can join multiple tables



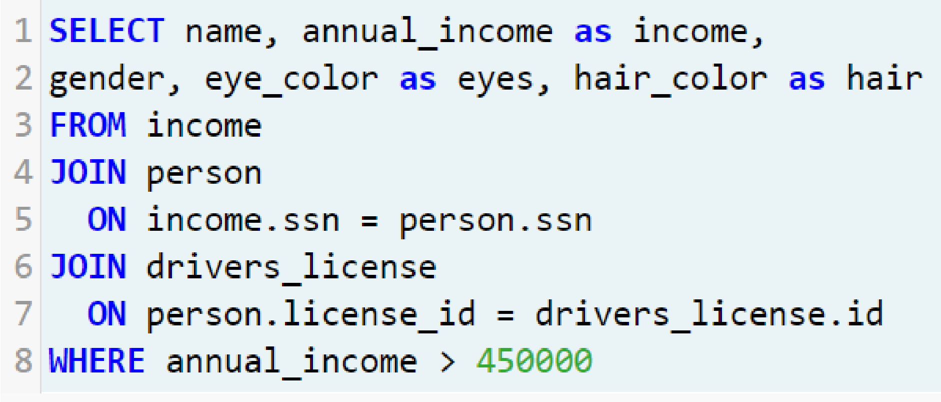
```
select person.name, person.ssn,
       income.ssn, annual_income
FROM income
JOIN person
ON income.ssn = person.ssn
WHERE annual_income > 450000
```



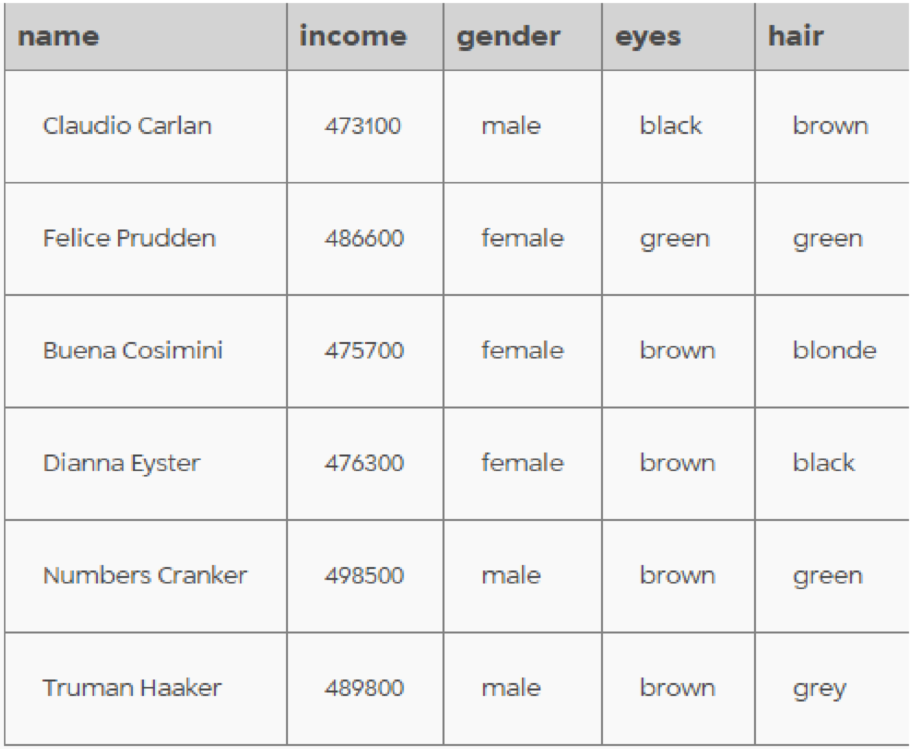
name	ssn	ssn	annual_income
Claudio Carlan	311494850	311494850	473100
Felice Prudden	118015315	118015315	486600
Buena Cosimini	313890530	313890530	475700
Dianna Eyster	541217354	541217354	476300
Numbers Cranker	361660921	361660921	498500
Truman Haaker	121635236	121635236	489800

JOINs with 2+ tables

Just keep adding JOIN <table name> ON <matching fields>
Test iteratively!



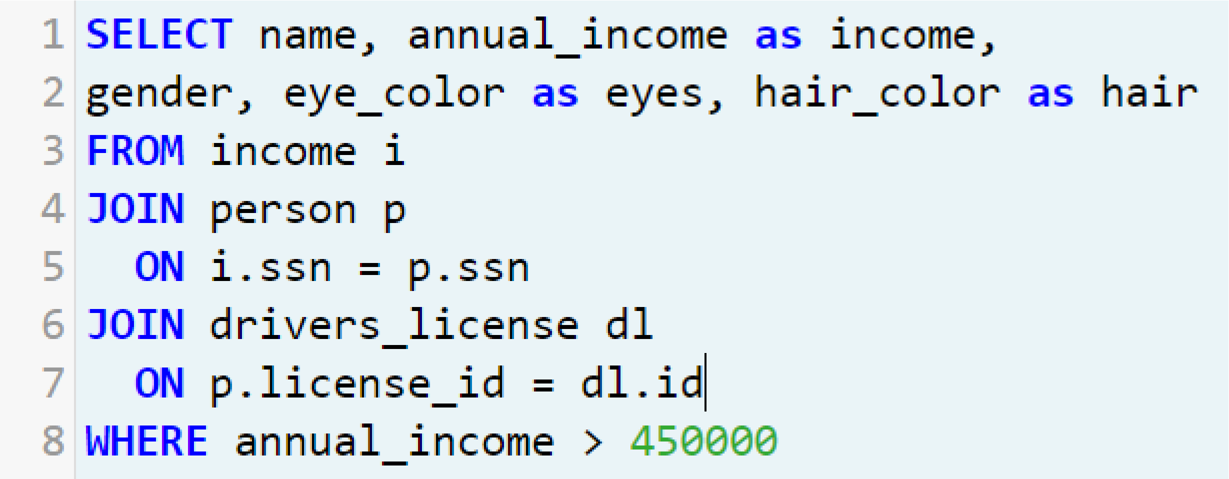
```sql
SELECT name, annual_income as income,
gender, eye_color as eyes, hair_color as hair
FROM income
JOIN person
  ON income.ssn = person.ssn
JOIN drivers_license
  ON person.license_id = drivers_license.id
WHERE annual_income > 450000
```



name	income	gender	eyes	hair
Claudio Carlan	473100	male	black	brown
Felice Prudden	486600	female	green	green
Buena Cosimini	475700	female	brown	blonde
Dianna Eyster	476300	female	brown	black
Numbers Cranker	498500	male	brown	green
Truman Haaker	489800	male	brown	grey

Aliases

- Lets you more easily refer to columns in a long query.
- A column is given an alias for the query (annual_income as income)
- A table is given an alias (person as ‘p’)

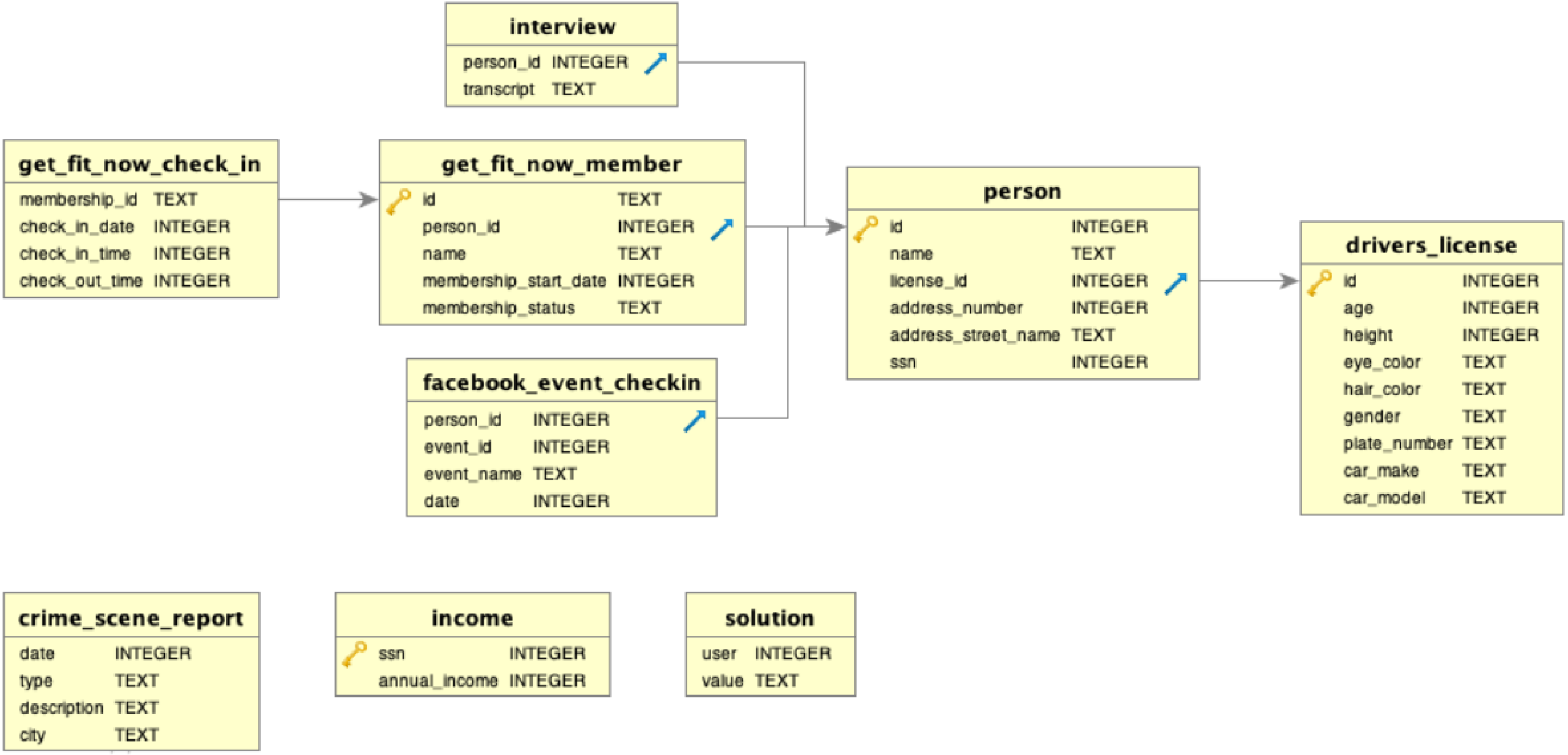


```
SELECT name, annual_income as income,
gender, eye_color as eyes, hair_color as hair
FROM income i
JOIN person p
  ON i.ssn = p.ssn
JOIN drivers_license dl
  ON p.license_id = dl.id
WHERE annual_income > 450000
```

Let’s Solve the Murder

- Review the facts
- Ask the right questions
- Solve the puzzle

ERD Diagram



interview

person_id	INTEGER	↗
transcript	TEXT	

get_fit_now_check_in

membership_id	TEXT
check_in_date	INTEGER
check_in_time	INTEGER
check_out_time	INTEGER

get_fit_now_member

🔑	id	TEXT	
	person_id	INTEGER	↗
	name	TEXT	
	membership_start_date	INTEGER	
	membership_status	TEXT	

facebook_event_checkin

person_id	INTEGER	↗
event_id	INTEGER	
event_name	TEXT	
date	INTEGER	

person

🔑	id	INTEGER	
	name	TEXT	
	license_id	INTEGER	↗
	address_number	INTEGER	
	address_street_name	TEXT	
	ssn	INTEGER	

drivers_license

🔑	id	INTEGER
	age	INTEGER
	height	INTEGER
	eye_color	TEXT
	hair_color	TEXT
	gender	TEXT
	plate_number	TEXT
	car_make	TEXT
	car_model	TEXT

crime_scene_report

date	INTEGER
type	TEXT
description	TEXT
city	TEXT

income

🔑	ssn	INTEGER
	annual_income	INTEGER

solution

user	INTEGER
value	TEXT

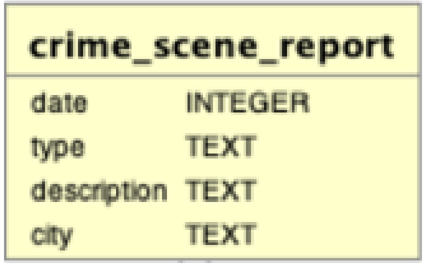
Notes: This database is design for learning purposes, and is not fully optimized (1NF, 2NF, etc)

SQL Murder Mystery

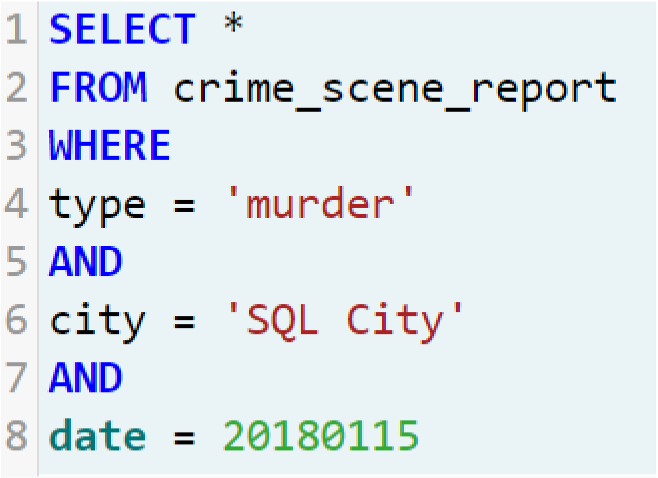
- A crime has taken place and the detective needs your help.
- The detective gave you the crime scene report, but you somehow lost it.
- You vaguely remember that the crime was a **murder** that occurred sometime on **Jan.15, 2018** and that it took place in **SQL City**.
- Start by retrieving the corresponding crime scene report from the police department's database.

Start the investigation

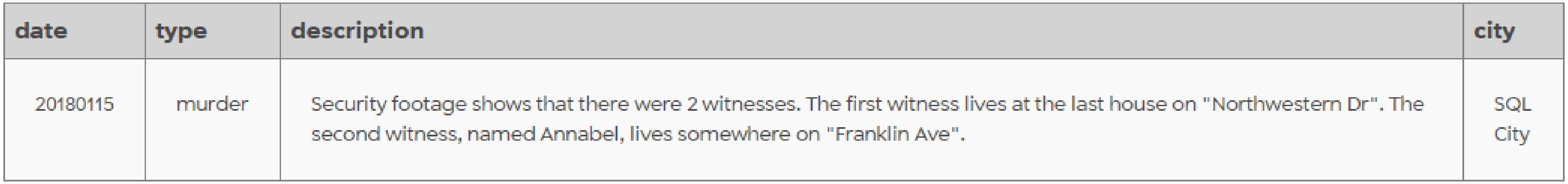
- crime_scene_report
  - Type = murder
  - Date = 01/15/2018
  - City = SQL City



crime_scene_report	
date	INTEGER
type	TEXT
description	TEXT
city	TEXT

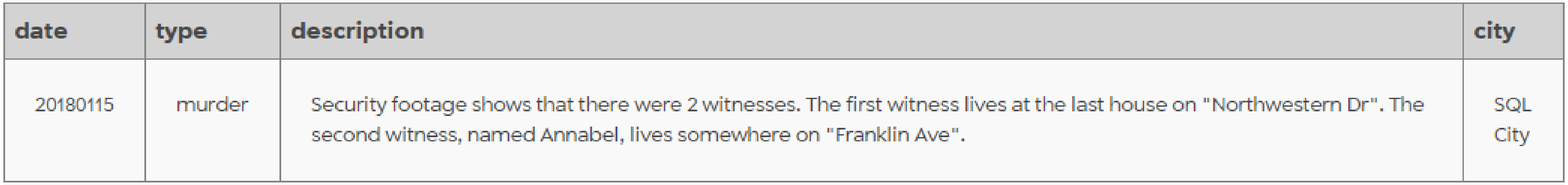


```sql
SELECT *
FROM crime_scene_report
WHERE
type = 'murder'
AND
city = 'SQL City'
AND
date = 20180115
```

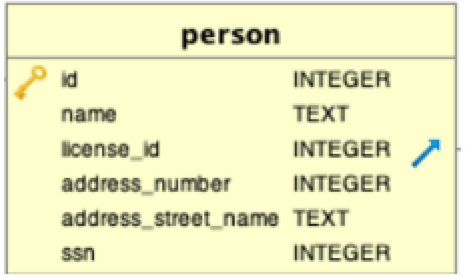


date	type	description	city
20180115	murder	Security footage shows that there were 2 witnesses. The first witness lives at the last house on "Northwestern Dr". The second witness, named Annabel, lives somewhere on "Franklin Ave".	SQL City

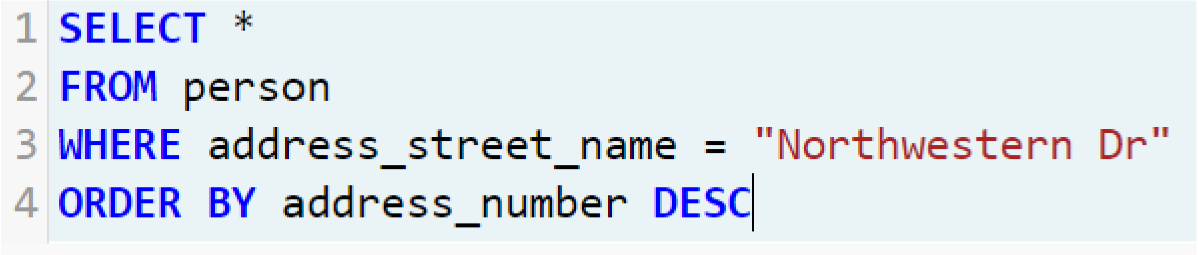
Witness #1



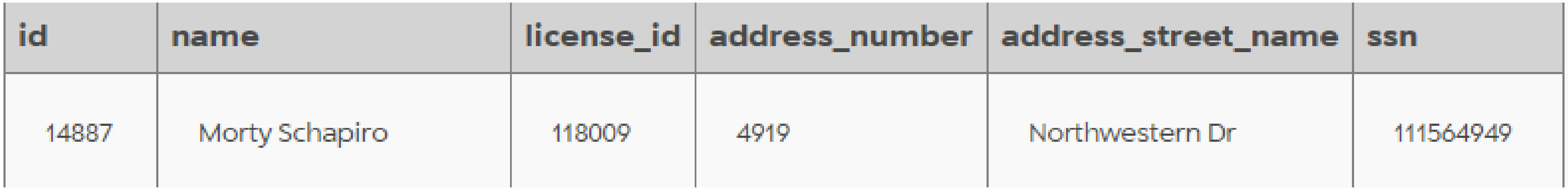
date	type	description	city
20180115	murder	Security footage shows that there were 2 witnesses. The first witness lives at the last house on "Northwestern Dr". The second witness, named Annabel, lives somewhere on "Franklin Ave".	SQL City



person	
id	INTEGER
name	TEXT
license_id	INTEGER
address_number	INTEGER
address_street_name	TEXT
ssn	INTEGER



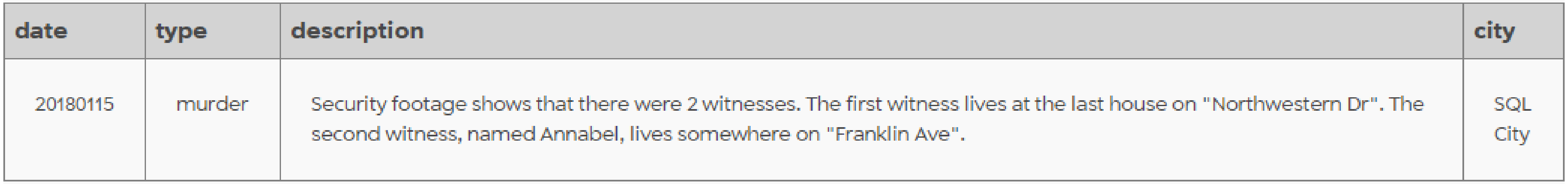
```
SELECT *
FROM person
WHERE address_street_name = "Northwestern Dr"
ORDER BY address_number DESC
```



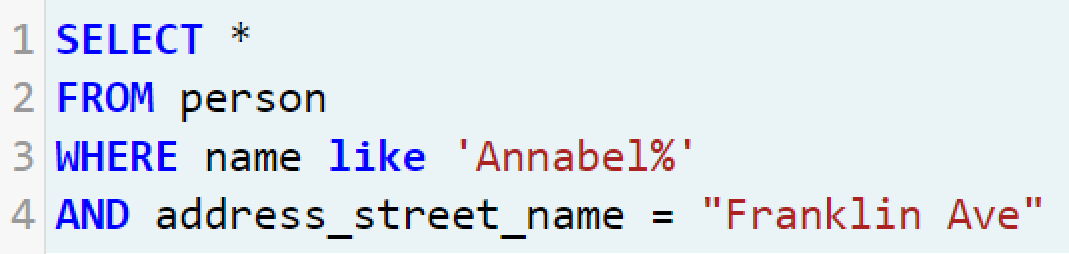
id	name	license_id	address_number	address_street_name	ssn
14887	Morty Schapiro	118009	4919	Northwestern Dr	111564949

Note how the DESC modifier returns the results in reverse, or descending, order

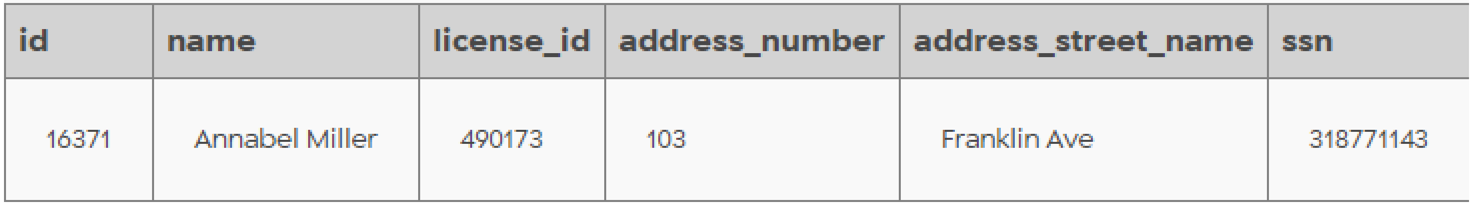
Witness #2



date	type	description	city
20180115	murder	Security footage shows that there were 2 witnesses. The first witness lives at the last house on "Northwestern Dr". The second witness, named Annabel, lives somewhere on "Franklin Ave".	SQL City

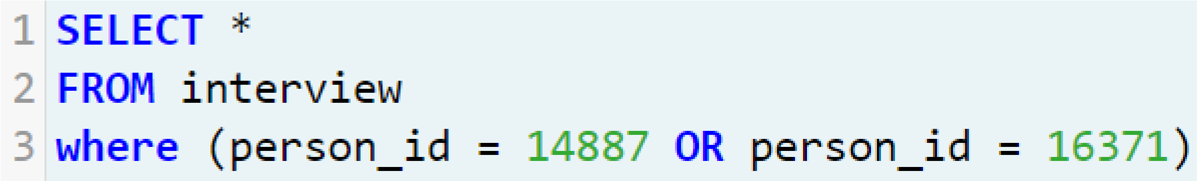


```sql
SELECT *
FROM person
WHERE name like 'Annabel%'
AND address_street_name = "Franklin Ave"
```

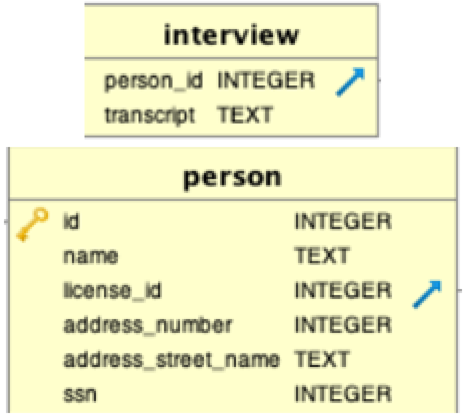


id	name	license_id	address_number	address_street_name	ssn
16371	Annabel Miller	490173	103	Franklin Ave	318771143

Access the Witness Statements



```sql
SELECT *
FROM interview
where (person_id = 14887 OR person_id = 16371)
```

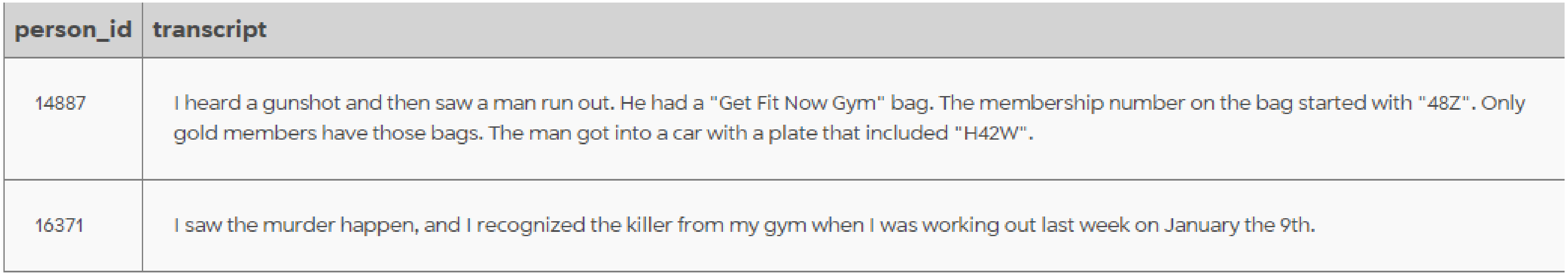


interview

person_id	INTEGER
transcript	TEXT

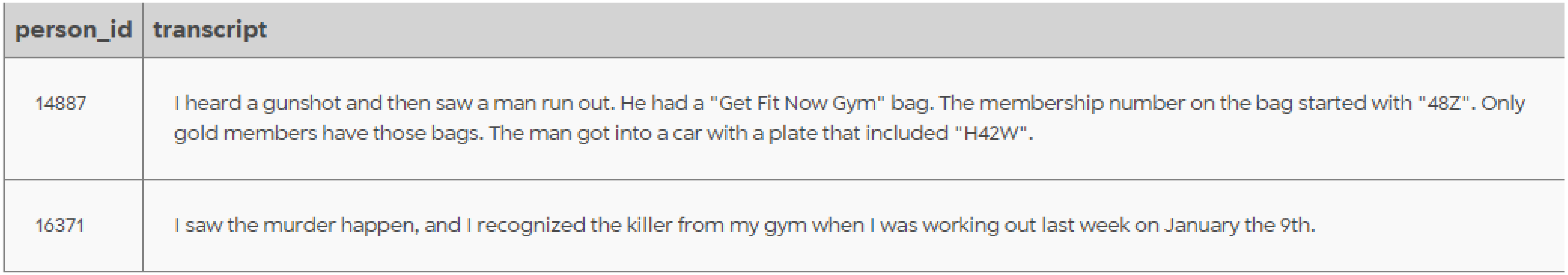
person

id	INTEGER
name	TEXT
license_id	INTEGER
address_number	INTEGER
address_street_name	TEXT
ssn	INTEGER



person_id	transcript
14887	I heard a gunshot and then saw a man run out. He had a "Get Fit Now Gym" bag. The membership number on the bag started with "48Z". Only gold members have those bags. The man got into a car with a plate that included "H42W".
16371	I saw the murder happen, and I recognized the killer from my gym when I was working out last week on January the 9th.

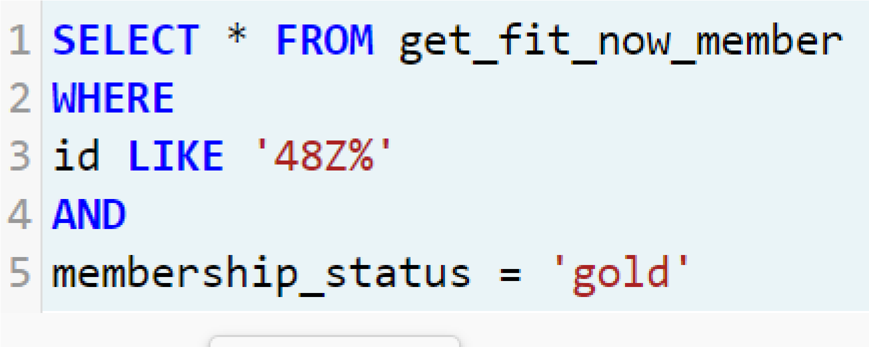
Clues



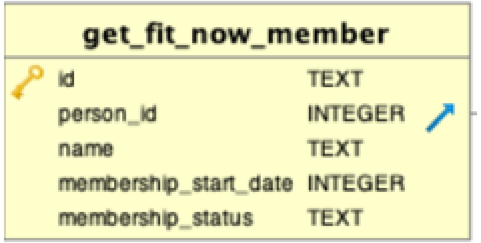
person_id	transcript
14887	I heard a gunshot and then saw a man run out. He had a "Get Fit Now Gym" bag. The membership number on the bag started with "48Z". Only gold members have those bags. The man got into a car with a plate that included "H42W".
16371	I saw the murder happen, and I recognized the killer from my gym when I was working out last week on January the 9th.

- It was a man
- He had a gym bag
- Gym Membership ID began with ‘48Z’
- Only Gold Members have that type of bag
- License plate of getaway car includes ‘H42W’
- Murder date was 2018-01-09

Select with compound Where clause

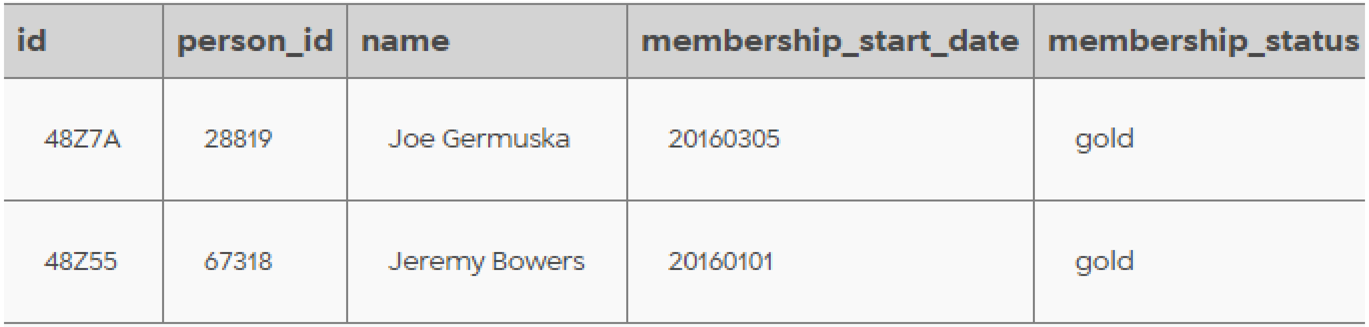


```sql
SELECT * FROM get_fit_now_member
WHERE
id LIKE '48Z%'
AND
membership_status = 'gold'
```



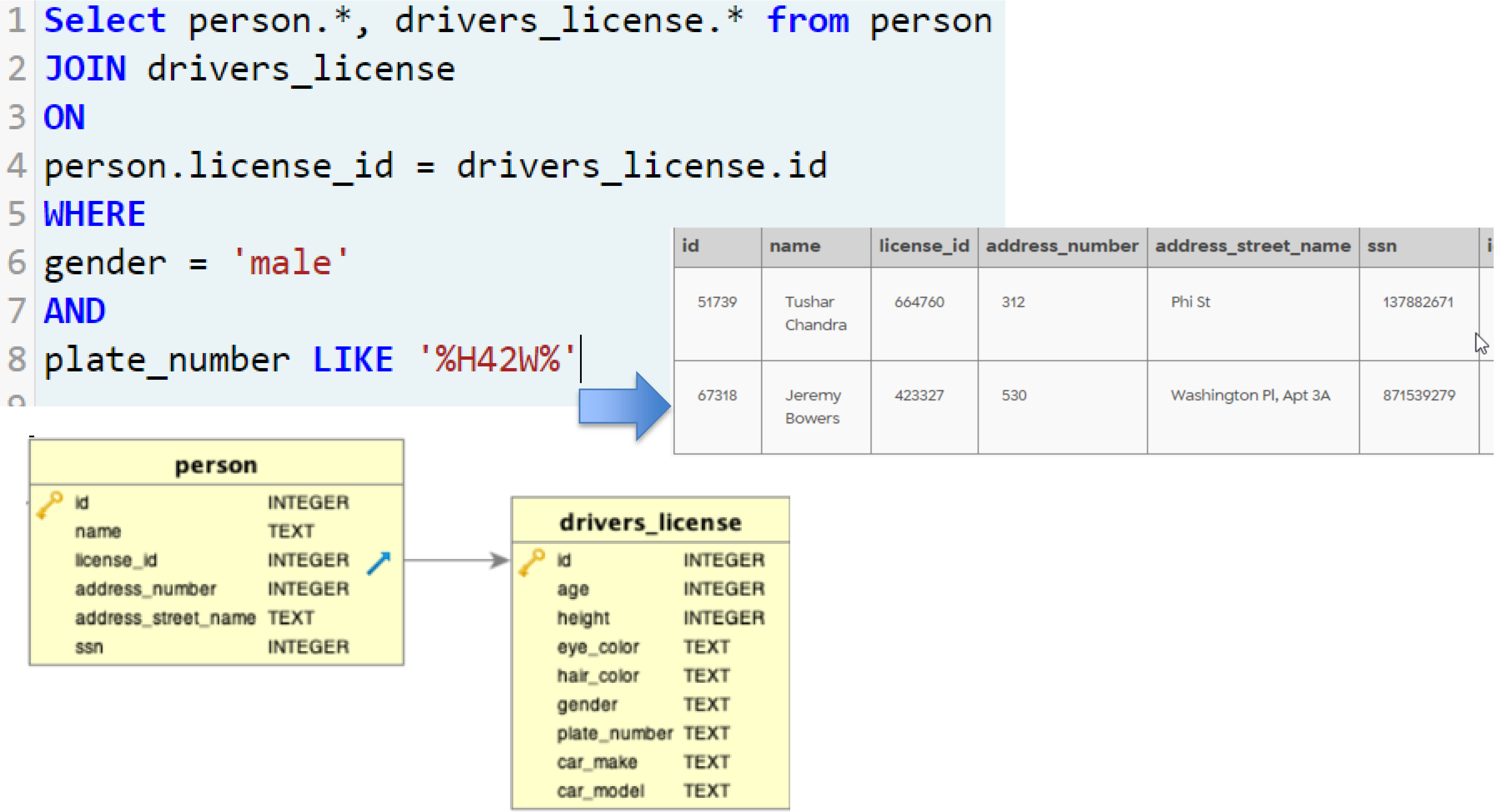
get_fit_now_member

id	TEXT
person_id	INTEGER
name	TEXT
membership_start_date	INTEGER
membership_status	TEXT



id	person_id	name	membership_start_date	membership_status
48Z7A	28819	Joe Germuska	20160305	gold
48Z55	67318	Jeremy Bowers	20160101	gold

Select with Join



```
Select person.*, drivers_license.* from person
JOIN drivers_license
ON
person.license_id = drivers_license.id
WHERE
gender = 'male'
AND
plate_number LIKE '%H42W%'
```

id	name	license_id	address_number	address_street_name	ssn	i
51739	Tushar Chandra	664760	312	Phi St	137882671	
67318	Jeremy Bowers	423327	530	Washington Pl, Apt 3A	871539279	

person	
id	INTEGER
name	TEXT
license_id	INTEGER
address_number	INTEGER
address_street_name	TEXT
ssn	INTEGER

drivers_license	
id	INTEGER
age	INTEGER
height	INTEGER
eye_color	TEXT
hair_color	TEXT
gender	TEXT
plate_number	TEXT
car_make	TEXT
car_model	TEXT